Managing Macros with the Visual Basic Editor

By Paul Cornell

In this month's column, I show you how to get the maximum benefit out of the Microsoft Visual Basic® Editor, which is a tool you can use to manage your macros.

See all Power User columns
See all columns

Office Help
Office Quiz
Office Colu
Spotlight A
Find an Of
Print Posta

Web Pa
Dow

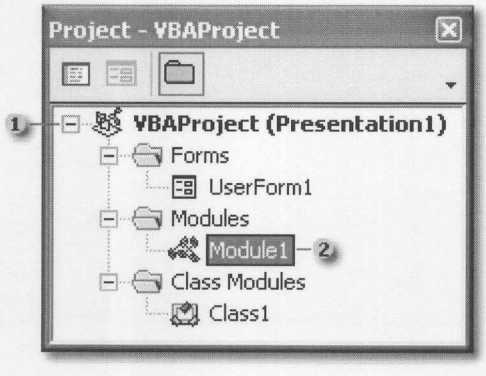
Print pos
from Ou

Get FR
Hel

If you have read the previous Office Power User Corner columns, you know how to create and record your own Microsoft Office macros. Macros, as you recall, are code components that automate repetitive Office tasks.

What is the Visual Basic Editor?

The Visual Basic Editor, included with most Office applications, is the environment you use to create, modify, and manage Office macros. A macro commonly consists of code starting with the keyword `Sub` and ending with the keywords `End Sub`. (This code is also known as a **subroutine**). A **module** contains one or more macros or subroutines, while a **project** contains one or more modules.



1 A project can contain one or more modules.
2 A module can contain one or more macros or subroutines.

The Visual Basic Editor is used to manage projects and their associated modules. To access the Visual Basic Editor, on the **Tools** menu, point to **Macro** and then click **Visual Basic Editor**.

Try it out!

Let's create a simple macro and then use the Visual Basic Editor to modify our macro later. The macro we'll create changes the background color of a

Microsoft Word document based on the color values you specify. Here's how to create and run the macro:

1. Start Microsoft Word.
2. Create a new blank document.
3. On the **Tools** menu, point to **Macro** and then click **Macros**.
4. In the **Macros in** box, click **Normal.dot (global template)**.
5. In the **Macro name** box, type **ChangeColor** and then click **Create**. The following code appears in the Visual Basic Editor:

```
Sub ChangeColor()
    '
    ' ChangeColor Macro
    ' Macro created {date} by {name}
    '

End Sub
```

6. Next, edit the code so that it looks like this:

```
Sub ChangeColor()
    '
    ' ChangeColor Macro
    ' Macro created {date} by {name}
    '

    Dim intRed As Integer
    Dim intGreen As Integer
    Dim intBlue As Integer

    intRed = InputBox("Red value? (0-255)")
    intGreen = InputBox("Green value? (0-255)")
    intBlue = InputBox("Blue value? (0-255)")

    ActiveWindow.View = wdWebView

    With ActiveDocument.Background.Fill

        .Visible = msoTrue
        .ForeColor.RGB = RGB(intRed, intGreen, intBlue)

    End With

    MsgBox "The document's background is now RGB(" & _
        intRed & ", " & intGreen & ", " & intBlue & ")."

End Sub
```

7. On the **View** menu, click **Microsoft Word**.
8. Type some text in the document so you can see the contrast between the document's background and the text.
9. On the **Tools** menu in Word, point to **Macro**, and click **Macros**.
10. Click **ChangeColor**, and click **Run**.
    Note If a message appears stating that the macros in the project are disabled, do one or both of the following:
    - Refer to the Microsoft Product Support Services article WD2002: "The Macros in the Project Are Disabled" Message When You Run a Macro (Q290949) to enable the macro to run properly.
    - Refer to the assistance article Changing Macro Security Settings in Office XP.
11. In the dialog box that appears, enter a whole number between 0 and 255 three times (for each value of red, green, and blue), and click **OK** each time.

The document's background color changes to match the combined red-

green-blue (RGB) value you provided in the previous step.

Now, let's use the Visual Basic Editor to modify the macro. Let's change the background color of all of the text in the Word document based on the color values you specify. To do this:

1. On the **Tools** menu, point to **Macro** and click **Visual Basic Editor**.
2. In the **Project** window, expand the **Normal** folder if it is not already expanded.
   Note If the **Project Explorer** window is not visible, on the **View** menu, click **Project Explorer**.
3. In the expanded **Normal** folder, expand the **Modules** folder if it is not already expanded.
4. Double-click **NewMacros**.
5. Locate the **ChangeColor** macro code from the previous procedure.
   Note If you have problems locating the **ChangeColor** macro, try clicking **ChangeColor** in the **Procedure** dropdown list at the top right edge of the **Code** window.
6. Next, edit the code so that it looks like this (changes from the previous code are highlighted in **bold**):

```
Sub ChangeColor()
'
' ChangeColor Macro
' Macro created {date} by {name}
'

    Dim intRed As Integer
    Dim intGreen As Integer
    Dim intBlue As Integer

    intRed = InputBox("Red value? (0-255)")
    intGreen = InputBox("Green value? (0-255)")
    intBlue = InputBox("Blue value? (0-255)")

    ActiveWindow.View = wdWebView

    With ActiveDocument

        .Select
        .Range.Font.Color = RGB(intRed, intGreen, intBlue)

    End With

    MsgBox "The document's text is now RGB(" & _
        intRed & ", " & intGreen & ", " & intBlue & ")."

End Sub
```

7. Click anywhere in the **ChangeColor** macro between the `Sub` and `End Sub` keywords.
8. On the **Run** menu, click **Run Sub/UserForm**.
9. In the dialog box that appears, enter whole numbers between 0 and 255 for each value of red, green, and blue, and then click **OK** for each of the red, green, and blue input boxes that appear.

The document's text color changes to match the combined red-green-blue (RGB) value you provided in the previous step.

Managing modules and projects

The main code-management tool you use in the Visual Basic Editor to work with modules and projects is called the **Project Explorer**. The **Project** window is usually visible when you open the Visual Basic Editor. If the **Project Explorer** window is not visible, it can be displayed by clicking

Project Explorer on the **View** menu in the Visual Basic Editor.

The **Project** window displays the contents of all stored macro code associated with the active Office application and/or Office document. The following table lists the available macro code storage locations for the main Office applications.

Application	Available macro code storage locations
Microsoft Access 2002	Only within the database in which the code was created.
Microsoft Excel 2002	Only within the workbook in which the code was created.
Microsoft FrontPage® 2002	Only within the FrontPage application itself (in a file named FrontPage.fpm that resides on the local computer). You cannot store macro code within a particular FrontPage Web or within individual Web pages.
Microsoft Outlook® 2002	Only within the Outlook application itself (in a file named VbaProject.otm that resides on the local computer). You cannot store macro code within individual Outlook items.
Microsoft PowerPoint® 2002	Only within the presentation in which the code was created.
Microsoft Word 2002	With the Normal.dot template, any other available Word template, or the active document.

Sharing macros with others

When you distribute databases, spreadsheets, presentations, and documents containing macros to others, they can start running the attached macros immediately. You can also distribute individual modules containing macros, separate from Office documents and projects, to other Office users.

To distribute individual modules separate from Office documents and projects

1. Right-click the module containing your macro in the **Project** window.
2. Click **Export File** and follow the directions to save the exported code to a stand-alone file.

Note that the code in this file will not run on its own, however; it must be imported into an existing Office document or project, depending on the Office application.

To import a code file

1. On the **File** menu in the Visual Basic Editor, click **Import File**.
2. Follow the directions to bring the file's code into your document or project.

Next time

Now that we have most of the preliminary information about working with macros out of the way, I will start next time by sharing several macros that will help you speed up your work. Until then, experiment with the Visual Basic Editor and use the Visual Basic Editor's menu commands and shortcut key combinations to help you manage your macros.

Keep sending those e-mails!

I look forward to receiving your e-mails at pwruser@microsoft.com. I really want this to be your column, so please send me your comments and favorite handcrafted Office solutions. Remember, I will not be able to feature every Office solution that I receive, I will not have the time to respond to all of your e-mail, and I am not a technical support representative. But I may feature your solution in an upcoming column.

About the author

Paul Cornell works for the Office Help team. In addition to writing the Office Power User Corner column, Paul writes the Office Talk column for the Microsoft Developer Network (MSDN).

If you like this column and want to hear about more fun and useful Office offerings, sign up for our newsletter.

See all Power User columns
See all columns

Back to top